Inversion of Heavy Current Electroheat Problems on a Graphics Processing Unit (GPU)

Victor U. Karthik[1], Sivamayam Sivasuthan[1,] Arunasalam Rahunanthan[2], Ravi S. Thyagarajan[3] and Paramsothy Jayakumar[3], Lalita Udpa[1] and S. Ratnajeevan H. Hoole[1]*

1. Dept. of Electrical and Computer Engineering, Michigan State University, East Lansing, MI 48824, USA.
2. Department of Mathematics and Computer Science, Edinboro University of Pennsylvania, Edinboro, PA 16444, USA.
3. The US Army Tank Automotive Research, Development and Engineering Center, Warren, MI 48397-5000, USA.

Email addresses in order: uthayaku@msu.edu, sivasuth@msu.edu, rahunanthana@gmail.com, ravi.s.thyagarajan.civ@mail.mil, paramsothy.jayakumar.civ@mail.mil, udpal@msu.edu, srhhoole@gmail.com

*Correspondence

ABSTRACT: The inversion of electroheat problems is important in electrical machine design, metallurgical processes of mixing, and hyperthermia treatment in oncology. One of the important computations involves synthesizing the electromagnetic arrangement of coils so as to accomplish a desired heat distribution to achieve mixing, reduce machine heat or burn cancerous tissue. Two finite element problems need to be solved, first for the magnetic fields and the joule heat that the associated eddy currents generate and then, based on these heat sources, the second finite element problem for heat distribution. This two part problem needs to be iterated on to obtain the desired thermal distribution by optimization. This represents a heavy computational load associated with long wait-times before results are ready. The graphics processing unit (GPU) has recently been demonstrated to enhance the efficiency of the finite element field computations and cut down solution times. In this paper, given the heavy computational load from the two-part problem and the optimization, we use the GPU to perform the electroheat optimization by the genetic algorithm to achieve computational efficiencies better than those reported for a single finite element problem. The feasibility of the method is established through the simple problem of shaping a current carrying conductor so as to yield a constant temperature along a line.

Keywords: GPU, Multi-physics, FEM Optimization.

Optimization of Electro-Heat Systems for Desired Temperature

Heavy currents always lead to heating through the joule effect. This heat is often undesirable as in electrical machinery like alternators where the heat not only diminishes the efficiency of the generator but also can damage the insulation [1]. In other cases this heat can be beneficial as in a) the metallurgical industry where the heat is used to melt the ore and mix it through electromechanical forces [2-4] or b) hyperthermia treatment in oncology where cancerous tissue is burnt off although with lower currents, achieving the heating by stronger eddy currents through

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **10 NOV 2013**	2. REPORT TYPE **Journal Article**	3. DATES COVERED **09-04-2013 to 13-07-2013**
4. TITLE AND SUBTITLE **Inversion of Heavy Current Electroheat Problems on a Graphics Processing Unit (GPU)**		5a. CONTRACT NUMBER
		5b. GRANT NUMBER
		5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S) **Victor Karthik; Sivamayam Sivasuthan; Arunasalam Rahunanthan; Ravi Thyagarajan; Paramsothy Jayakumar**		5d. PROJECT NUMBER
		5e. TASK NUMBER
		5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Dept. of Electrical and Computer Engineering,Michigan State University,East Lansing,Mi,48224**		8. PERFORMING ORGANIZATION REPORT NUMBER **; #24316**
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) **U.S. Army TARDEC, 6501 East Eleven Mile Rd, Warren, Mi, 48397-5000**		10. SPONSOR/MONITOR'S ACRONYM(S) **TARDEC**
		11. SPONSOR/MONITOR'S REPORT NUMBER(S) **#24316**

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
Submitted to journal Mathematics Problems in Engineering

14. ABSTRACT
The inversion of electroheat problems is important in electrical machine design, metallurgical processes of mixing, and hyperthermia treatment in oncology. One of the important computations involves synthesizing the electromagnetic arrangement of coils so as to accomplish a desired heat distribution to achieve mixing, reduce machine heat or burn cancerous tissue. Two finite element problems need to be solved, first for the magnetic fields and the joule heat that the associated eddy currents generate and then, based on these heat sources, the second finite element problem for heat distribution. This two part problem needs to be iterated on to obtain the desired thermal distribution by optimization. This represents a heavy computational load associated with long wait-times before results are ready. The graphics processing unit (GPU) has recently been demonstrated to enhance the efficiency of the finite element field computations and cut down solution times. In this paper, given the heavy computational load from the two-part problem and the optimization, we use the GPU to perform the electroheat optimization by the genetic algorithm to achieve computational efficiencies better than those reported for a single finite element problem. The feasibility of the method is established through the simple problem of shaping a current carrying conductor so as to yield a constant temperature along a line.

15. SUBJECT TERMS
GPU, Multi-physics, FEM Optimization.

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Public Release**	**17**	

higher 1 kHz frequency [5, 6]. Whatever the situation it is often desirable to accomplish a particular thermal distribution – whether to save an alternator from overheating or to accomplish the necessary melting of the ore or to burn cancerous tissue without hurting healthy tissue.

As shown in Fig. 1, the design process involves setting the parameters$\{p\}$ that describe the electro-heat system (consisting of dimensions and material values), solving the eddy current problem for the magnetic vector potential A:

$$-\frac{1}{\mu}\nabla^2 A = J = \sigma_e E = \sigma_e[-j\omega A - \nabla\varphi] \tag{1}$$

where μ is the magnetic permeability, σ_e is the electrical conductivity, E the electrical field strength and $-\nabla\varphi$ is the externally imposed electric field driving the current [1,7]. Being a heavy current problem the frequency ω is low so that the current density J has only a conduction term $\sigma_e E$ and no displacement term $j\omega\epsilon E$. After finding A [8], we compute the joule heating density q from

$$E = -j\omega A - \nabla\varphi \tag{2}$$

and

$$q = \frac{\sigma_e}{2}E^2 \tag{3}$$

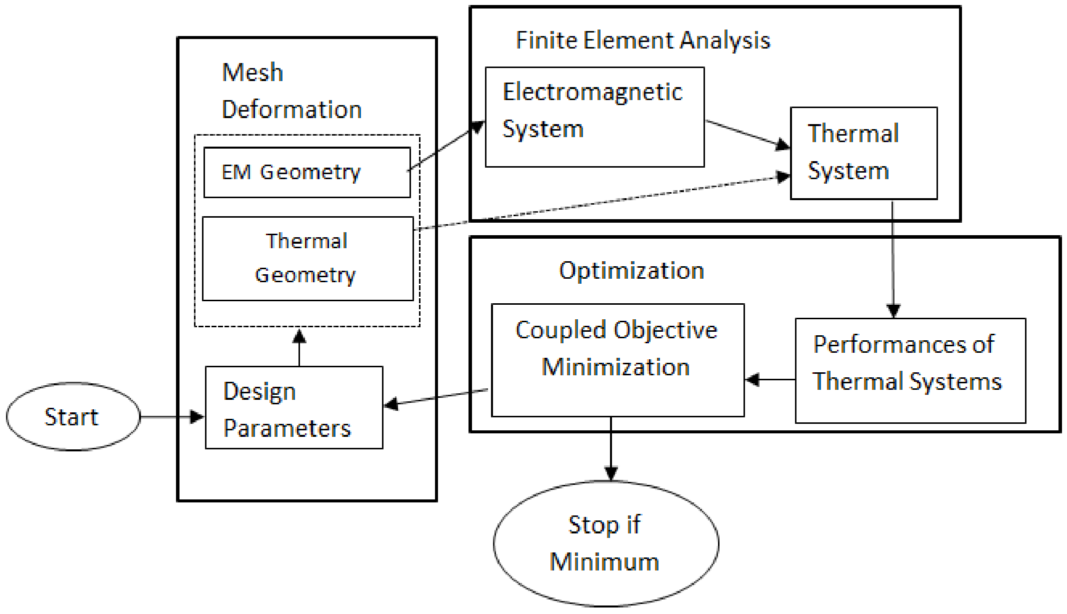


Figure 1: Finite Element Analysis and Optimization of Coupled Magneto-Thermal Problems

Once we have the heat source distribution q, the second problem of finding the resulting temperature is addressed by solving

$$-\sigma_t\nabla^2 T = q \tag{4}$$

where σ_t is the thermal conductivity [8]. Since the problem began with defining the parameters of system description {p}, we note that T = T({p}) since the computed T will depend on the values of {p}.

When a particular temperature distribution $T_0(x, y)$ is desired, the problem is one of finding that {p} which will yield

$$T(\{p\}) = T_o \qquad (5).$$

This is recognized as inverting (5) to find {p} and therefore referred to as the inverse problem which is now well understood in the literature, particularly when we are dealing with one branch of physics like electromagnetics [9-14].

In multi-branch coupled physics problems like the electroheat problem under discussion, {p} is defined in the electromagnetic system and F in the thermal system [8]. Further, when dealing with numerical methods such as the finite element method, T is given at the nodes (although technically T(x,y) may be derived from the finite element trial functions using the nodal values) and T_o, rather than being a function of x and y, is more conveniently defined at measuring points i, numbering say m. The design desideratum then may be cast as an object function F to be minimized with respect to the parameters {p}

$$F = F(\{p\}) = \frac{1}{2}\sum_{i=1}^{m}\left[T^i - T_0^i\right]^2 \qquad (6)$$

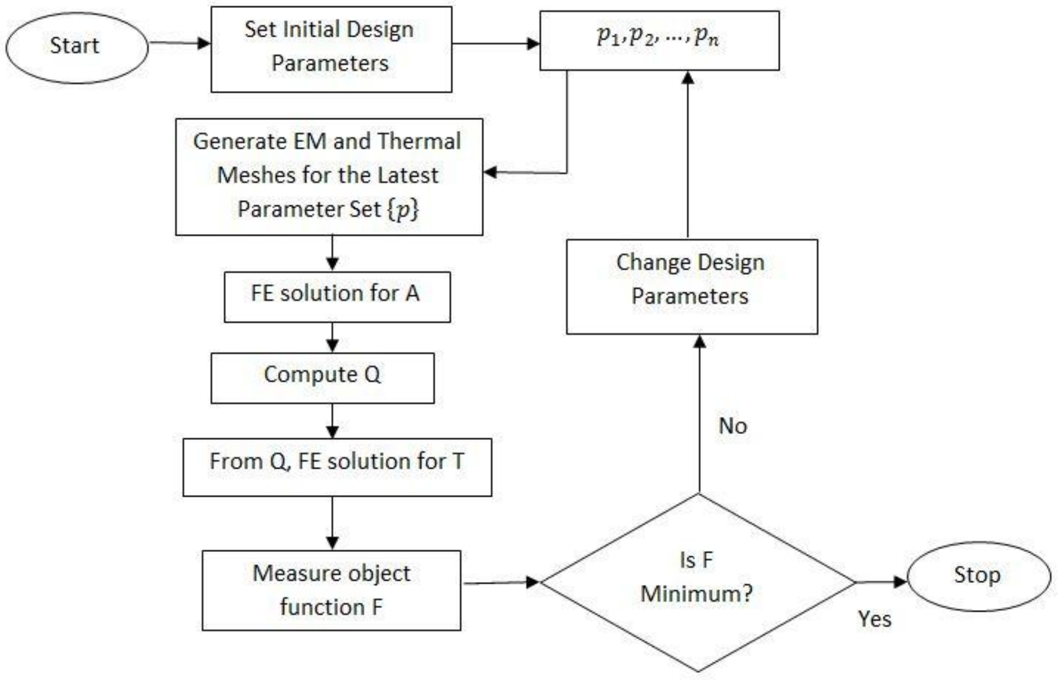

Start
Set Initial Design Parameters
$p_1, p_2, \dots, p_n$
Generate EM and Thermal Meshes for the Latest Parameter Set $\{p\}$
FE solution for A
Compute Q
From Q, FE solution for T
Measure object function F
Is F Minimum?
Yes
Stop
No
Change Design Parameters

Figure 2: The Design Cycle for the Coupled Electroheat Problem

where T^i is T as computed at the same m points i where T_0 is defined. The optimization process, by whatever method [15], keeps adjusting $\{p\}$ until F is minimized, making T^i come as close to T_0^i as it can – as close as it can rather than exactly to T_0^i because our design goal as expressed in (6) may not always be realistic and achievable. At that point $\{p\}$ would represent our best design. This specific process as it relates to Fig. 1, is shown in Fig. 2.

Optimization of Two-Physics Problems against Single Physics Problems

In the optimization of a single physics problem as in magnetostatics [9, 11-14], we construct one finite element mesh, solve for the magnetic vector potential A, and then change $\{p\}$. The method by which we change $\{p\}$ depends on the method of optimization we employ [15]. In coupled problems like the electroheat problem under discussion, two different meshes are often required [8]. For example, at a copper-air boundary in magnetics, both regions are nonmagnetic and therefore have the same permeability, the permeability of free space μ_0. However, for the thermal problem, they need to be modeled as two different regions because air has little thermal conductivity whereas copper is highly conductive.

Moreover, the optimization process too imposes huge difficulties depending on the method employed. In the simpler zeroth order methods, only the value of F, given $\{p,\}$ needs to be computed. This takes simply a finite element solution for a mesh constructed for the present value of $\{p\}$. In the more powerful first order methods, however,

$$\nabla F = \frac{\partial F}{\partial\{p\}} = \left\{\frac{\partial F}{\partial p_i}\right\} \tag{7}$$

needs to be computed [15]. This may be by finite differences – that is, to get the derivative of F with respect to p_i, we need to evaluate F(p_i), then adjust $p_i$by a very small amount δp_i, redo the finite element solution (which means a new mesh has to be generated for the changed geometry), and then evaluate F, which would be $F(p_i + \delta p_i)$. This gives us δF, thereby leading to the derivate by finite difference, $\delta F/\delta p_i$. This is computationally expensive because of having to be done for all m parameters p_i, which means the first finite element solution at $\{p\}$ and evaluation of F($\{p\}$) followed by m finite element solutions with only a particular p_i adjusted by δp_i and then the evaluation of m values of δF. That is, m+1 finite element meshes and solutions are required at great cost. Furthermore, even after all that work, it is known that the derivative by this finite difference process has poor accuracy [16]. The correct way to obtain accurate derivatives is from the derivative information inherent to the finite element solution through the finite element trial function; that is, although the solution for A is explicitly in terms of the values of A at the finite element nodes, we are really solving for A(x,y) as given by the trial function which is expressed through the nodal values of A [7]. In solving the finite element Dirichlet matrix equation

$$[P]\{A\} = \{Q\} \tag{8}$$

Although we explicitly solve for the nodal values $\{A\}$, it is really for the trial function A(x,y) that we are solving. Both the Dirichlet matrix [P] and right hand side $\{Q\}$in (8) are expressed as known functions of the vertex coordinates of the finite elements of the mesh, permeability, and current density. After solving for $\{A\}$, differentiating the equation with respect to p_i we obtain [7, 8, 12-14]

$$[P]\frac{d\{A\}}{dp_i} = \frac{d\{Q\}}{dp_i} - \frac{d[P]}{dp_i}\{A\} \tag{9}$$

where {A} has already been solved for, and $d\{Q\}/dp_i$ and $d[P]/dp_i$ are computable. There is further computational efficiency to be reaped [17] by using the Cholesky factorization method of splitting [P] into its lower triangular and upper triangular Cholesky factors [L] and [U]

$$[L][U] = [P] \tag{10}$$

which for symmetric [P] as in both magnetic and thermal field problems reduces to

$$[L][L]^t = [P] \tag{11}$$

so that solving (8), in its new form

$$[\mathrm{L}][\mathrm{U}]\{\mathrm{A}\} = \{\mathrm{Q}\} \tag{12}$$

reduces to solving

$$[L]\{z\} = \{Q\} \tag{13}$$

For {z} where

$$[U]\{A\} = \{z\} \tag{14}$$

and then, having found {z}, solving (14) for {A}. The computational efficiency lies in the fact that the main work in Cholesky's scheme for matrix equation solution is in finding [L] by solving (11). Thereafter the forward elimination in solving (13) and back substitution in solving (14) are trivial. That is, once [L] and [U] = $[L]^t$ are in hand, solving (9) with the same coefficient matrix [P] as (8) for the m gradients $\frac{d\{A\}}{dp_i}$ is trivial since only forward elimination and back substitution are required.

Be that as it may, while solving (9) is trivial, forming (9) is not because computing $\partial\{Q\}/\partial p_i$ and $\partial[P]/\partial p_i$ to form (9) is in terms of programming an arduous task that is not easily amenable to building up as general purpose software. As a particular $p_i$changes, some vertices of a few triangles will move [9, 14] and the analyst needs to keep track of whether one, two or all three of the vertices of a triangle move by δp_i. Very complex coding is required that is problem-specific rather than general purpose. In a coupled problem, computing the derivatives of the finite element equations for temperature with respect to parameters in the magnetic problem is prohibitively complex although there are ways around it [18, 19]. Programming the problem specific computations of $\partial\{Q\}/\partial p_i$ and $\partial[P]/\partial p_i$ is ill-advised because of the complexities.

Therefore first order optimization methods, more slowly convergent than gradient methods, are the best route to go for optimizing coupled electroheat problems. Simkin and Trowbridge [20] aver that simulated annealing and the evolution strategy (a variant of the genetic algorithm [21]) take many more function evolutions. Although they are computationally intensive they are far easier to implement, especially as general purpose software. Indeed commercial codes that need to be general purpose use zeroth order methods not necessarily because they are superior to gradient methods, but because once a finite element analysis program is developed by a company, giving it optimization capabilities only takes coupling it with an optimization package to which object function evaluations can be fed – whereas feeding both the object function and its gradient as would be required when gradient methods are in use would take extensive code development. In the context of single physics problems, Haupt [22] advises that the genetic algorithm is best for many discrete parameters and the gradient methods for where there are but a few continuous parameters. We rationalize this position on the grounds that gradient computation though difficult is more manageable when there are fewer parameters to optimize with respect to. Indeed, we have gone up to 30 continuous parameters using gradient methods without problems.

But we are now dealing with multi-physics electroheat problems to which these considerations based on single physics systems do not apply.

Method of Optimization – The Genetic Algorithm

Having settled on a zeroth order method of optimization that does not need gradient information on the object function, we take cognizance that zeroth order methods are statistical so that several-fold more object function computations need to be made. Random Search is too random in its searching and we need something more systematic in its searching the domain space so as to not lead to excessive computation times in relation to gradient methods.

The alternatives are simulated annealing and the genetic algorithm, both good methods widely used in industry and a part of commercial software [23]. Going by the literature Preis, Magele and Biro [24], staunch advocates of the zeroth order evolution strategy, merely say it is competitive with its higher order deterministic counterparts (which we take to mean the same in time at best), but claim its "robustness and generality" are superior. This we agree with because search methods will never see mesh-induced artificial local minima as a problem [25]. The weight of evidence seemed to favor the genetic algorithm over simulated annealing but not firmly so. So we did a quick study, the results of which, shown in Fig. 3, support the genetic algorithm. Therefore we chose the genetic algorithm, the features of which are shown in Fig. 4. We note that in results from other disciplines besides finite element optimization Manikas and Cain also say that "the genetic algorithm was shown to produce solutions equal to or better than simulated annealing" in their work [26].

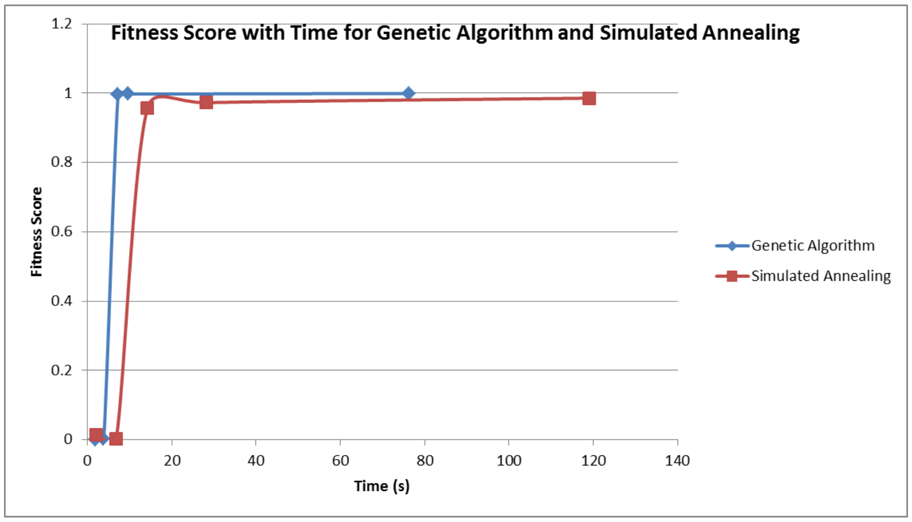


Figure 3: Genetic Algorithm Speed Compared with Simulated Annealing

In the genetic algorithm, the design parameter vector $\{p\}$ is represented by a binary encoding method. A chromosome is a vector $\{p\}$. Its fitness score f is defined in terms of the object function F. Although our object function F as defined in (6) is to be minimized, the fitness score f has to be maximized for the genetic algorithm. We therefore define the fitness score $f = \frac{1}{1+F}$

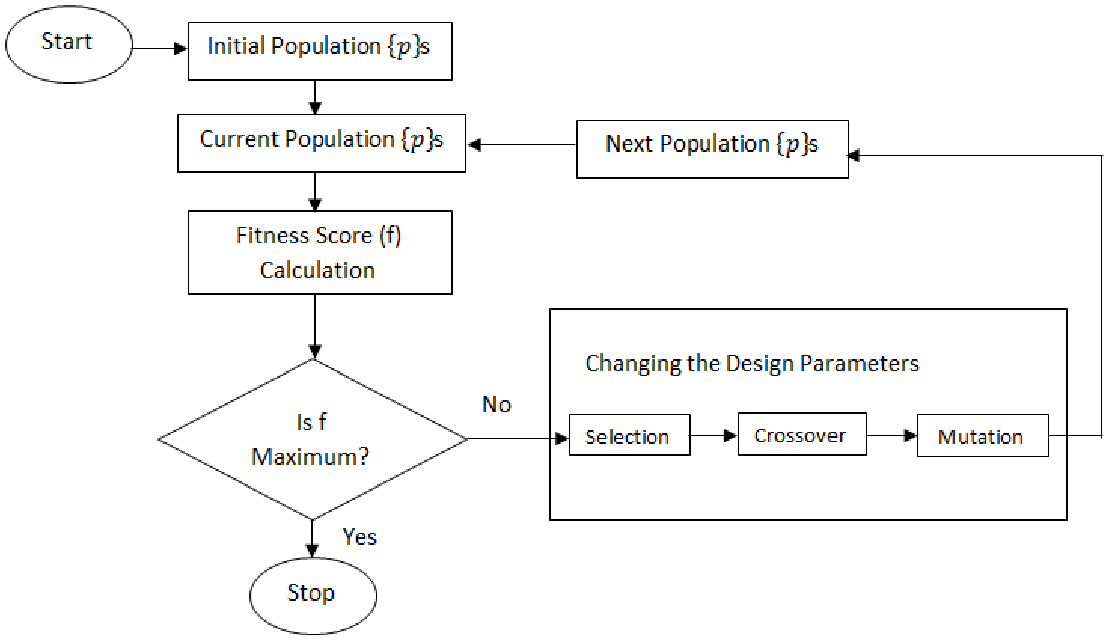

Start
Initial Population $\{p\}$s
Current Population $\{p\}$s
Next Population $\{p\}$s
Fitness Score (f) Calculation
Changing the Design Parameters
No
Selection
Crossover
Mutation
Is f Maximum?
Yes
Stop

Figure 4: Optimization Using the Genetic Algorithm

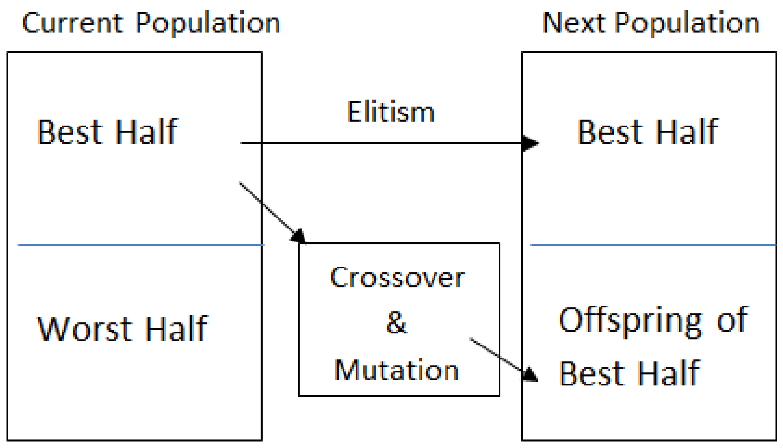


Figure 5.Changing design parameters in GA

(15)

Therefore when F goes to 0, f will reach it maximum value of 1. According to the methodology of optimization using GA as shown in Figs. 4 and 5, first we randomly generate hundreds of vectors $\{p\}$ (each called a chromosome) and this set is termed the initial population. With parallels to evolution, a new generation is to be created based on the best of this population. Then the fitness score for each {p} is calculated and checked as to whether there is a score at 1 or close enough for our purposes. This computation involves computing F according to (6) and therefore a finite element solution for that {p}. If there is no f satisfactorily close to 1, then the design parameters are changed according to the GA's classical way of selection, namely selection, crossover, and mutation (Fig. 4). In the selection process we use [23], selection of the new generation is based on elitism, which first copies the best half of the chromosomes to the new population without any changes (Fig. 5). Elitism can very rapidly increase the performance of GA because it prevents losing the best found solutions (meaning those sets {p} giving the best, i.e., highest, f values). The remaining half of the population will be replaced by offspring of the best half after crossover, and mutation as shown in Fig. 5. Here the crossover shown is one point crossover, whereby we randomly select one crossover point and then copy everything before this point from the first parent and then everything after the crossover point from the second parent. After a crossover is performed, mutation takes place. Mutation is an important part of the genetic search: it helps to prevent the population from stagnating at local optima. The mutation operator simply inverts the value of a randomly chosen gene of a chromosome. Now the current population will be replaced by the new population. This process will be repeated until the stopping conditions of Fig. 4 are satisfied.

GPU Computation for Genetic Algorithms for Electroheat Problems

We have at this point decided on optimization by the genetic algorithm in favor of other zeroth order methods and gradient methods. Naturally the work in two-physics electroheat problems in finite element GA optimization is far beyond that for a single finite element solution. First we have a 2-part coupled problem, having to solve for the magnetic field and then the thermal problem where we solve for the temperature. For realistic problems this has to be done several times – indeed tens of thousands of times – in searching the solution space for the minimum object function. Wait times can be excessive, making optimization practicably infeasible.

To cut down solution time, parallel processing needs to be resorted to [17, 27-32]. From the 1990s multiprocessor computers have been tried out. Typically with n processors (or computing elements), solution time could be cut down by almost a factor of (n-1) – (n-1) rather than n because one processor is reserved for controlling the other (n-1) and almost a factor of (n-1) rather than exactly (n-1) because of the additional operations of waiting while one processor accesses data being changed by another [27-32]. This route, however, is not desirable because supercomputers are prohibitively expensive and there are technical difficulties in sharing memory, because of which the available n values are usually 4, 8 and 16 and beyond that the costs become very high for normal engineering establishments and universities for broad acceptance.

Recently the graphics processing unit, endowed with much computing prowess to handle graphics operations, has been exploited to launch a computational kernel as several parallel threads [33]. This is ideally suited for object function evaluation as the kernel so that multiple threads can perform the finite element analyses and evaluation of f for each {p} in parallel. NVIDIA Corp's GPUs invented in 1999 and the Compute Unified Device Architecture (CUDA) [34] are today available on practically every PC as a standard and are increasingly exploited with more and more applications being ported thereto. Significantly the number of parallel threads is not limited as on a shared memory supercomputer.

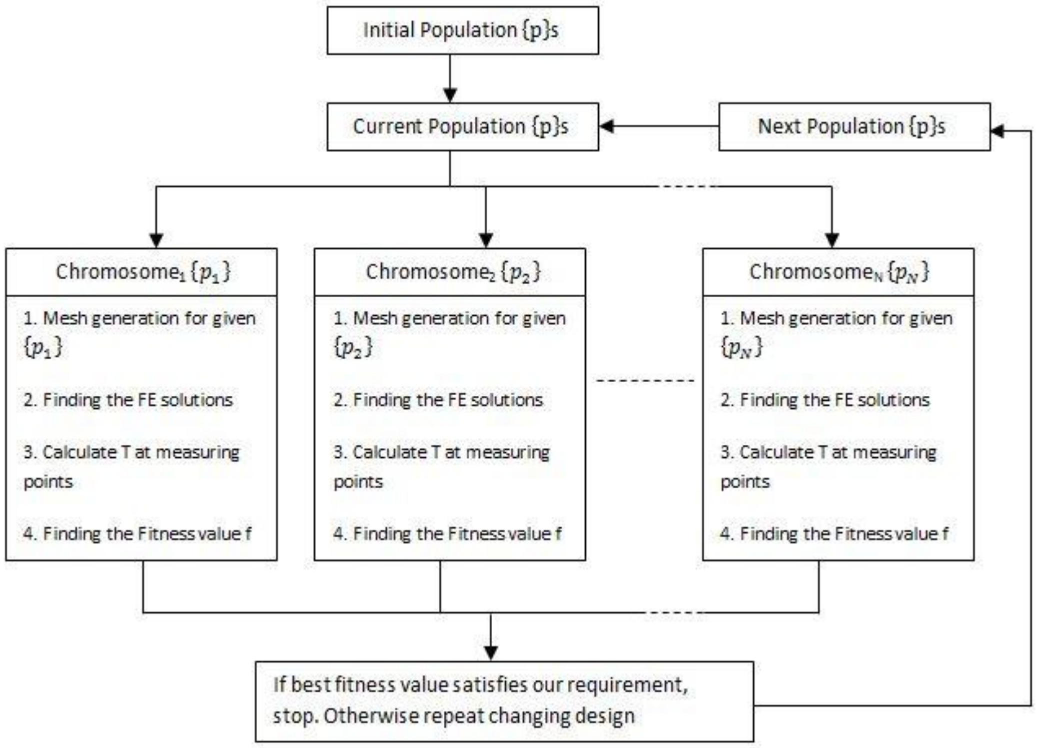
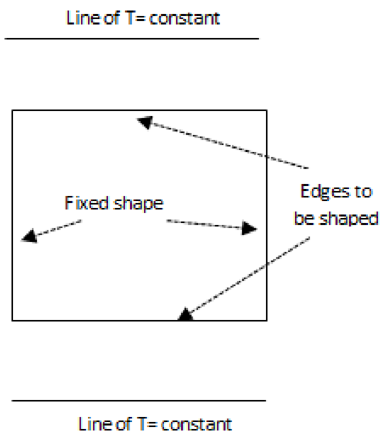

Initial Population {p}s
Current Population {p}s
Next Population {p}s
Chromosome$_1$ $\{p_1\}$
1. Mesh generation for given $\{p_1\}$
2. Finding the FE solutions
3. Calculate T at measuring points
4. Finding the Fitness value f
Chromosome$_2$ $\{p_2\}$
1. Mesh generation for given $\{p_2\}$
2. Finding the FE solutions
3. Calculate T at measuring points
4. Finding the Fitness value f
Chromosome$_N$ $\{p_N\}$
1. Mesh generation for given $\{p_N\}$
2. Finding the FE solutions
3. Calculate T at measuring points
4. Finding the Fitness value f
If best fitness value satisfies our requirement, stop. Otherwise repeat changing design

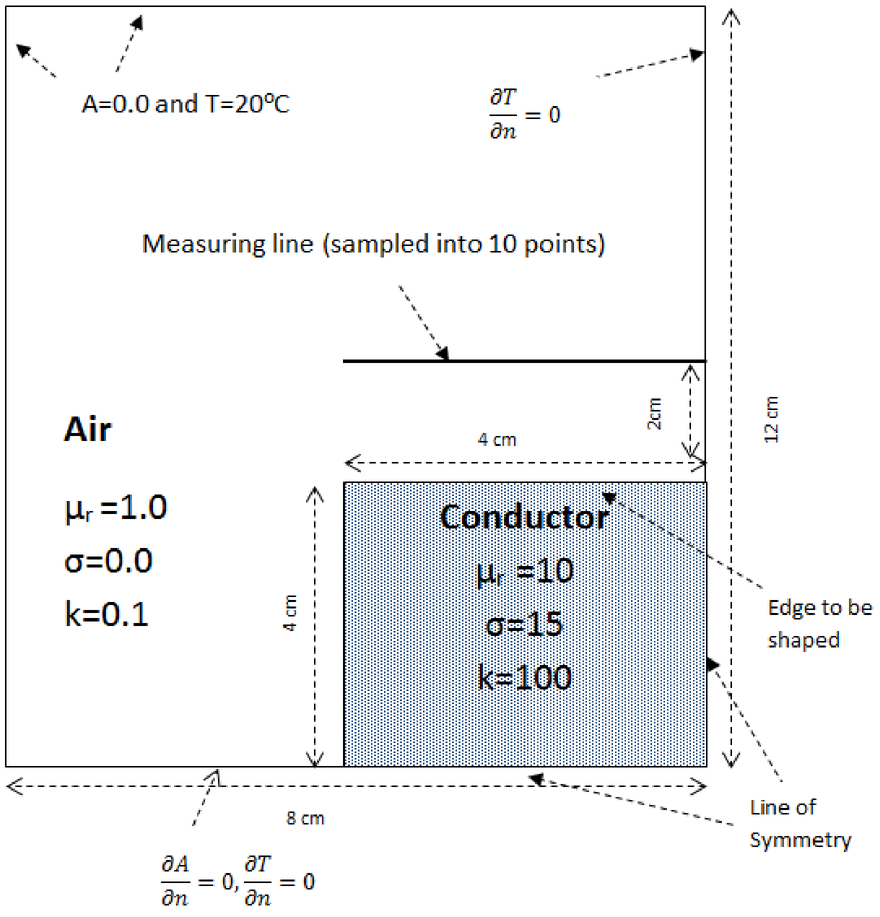
Figure 6: The Parallelized Process of the GPU

Cecka, Lew, and Darve [33] have also created and analyzed multiple approaches in assembling and solving sparse linear systems on unstructured meshes. The GPU coprocessor using single-precision arithmetic achieves speedups of 30 or so in comparison with a well optimized double-precision single core implementation [25]. We see that this is far better than the factor of just below 7 possible on a very expensive 8 processor supercomputer. So this is the way we will go, using the GPU to process the GA algorithm in parallel.

At present the limitations on GPU parallel processing are the 4 GB memory and the inability to launch a kernel in a parallel forks when that kernel is already a part of a parallelized fork – that is if we launch kernels as parallel thread with each thread doing a genetic algorithm evaluation, then within that kernel we cannot launch kernels for matrix solution in the manner of Cecka, Lew, and Darve [33]. But that is no real disadvantage because in the alternative of



a) Electrically Heated Conductor: The Actual Geometry


A=0.0 and T=20°C
$\frac{\partial T}{\partial n} = 0$
Measuring line (sampled into 10 points)
Air
μr =1.0
σ=0.0
k=0.1
2cm
12 cm
4 cm
4 cm
Conductor
μr =10
σ=15
k=100
Edge to be shaped
8 cm
Line of Symmetry
$\frac{\partial A}{\partial n} = 0, \frac{\partial T}{\partial n} = 0$

b) Symmetric Quarter: Boundary Value Problem

Figure 7: Numerical Model for Coupled Electroheat Problem

using shared memory supercomputer, for all practical purposes the hardware costs are prohibitive. Keeping in mind that forking of an already forked process is not possible on a GPU, we fork the fitness value computations as in Fig.

6. Fitness value calculation is the time consuming part as it involves both mesh generation, and finite element calculation. This forms a kernel that will be launched in parallel threads.

Therefore we divide the GPU threads and blocks of the same number as the population size and compute f values simultaneously (Fig.6). Since we have 65,536 blocks (2^{16}) and 512 threads in a general GPU, we can go up to a population size of $65{,}536 \times 512 = 33{,}554{,}432$ using the one GPU card on a PC. Since we do not need such a large population size for effective optimization, this is not restrictive. All the finite element calculation parts are programmed on the GPU in the CUDA C language. So when, given each $\{p\}$, we launch the fitness computation kernel as several threads (one for each $\{p\}$), the fitness score for all chromosomes will be calculated at the same time in parallel. But for each chromosome, the finite element calculation will be done sequentially (Fig. 6) and not parallelized along the lines of Cecka, Lew, and Darve. [33] because that would be attempting to fork within a fork.

Test Problem: Shaping an Electroheated Conductor to Achieve Temperature Profile

The test problem chosen is a simple one on which the method can be demonstrated and its feasibility established. Shown in Fig. 7a is a rectangular conductor which is heated by a current through it. The equi-temperature profiles would be circle-like around the conductor. But we want a constant temperature along two lines parallel to the pre-shaping rectangular conductor's two opposite edges to be shaped (Fig. 7). The question is this: how should that edge be reshaped to accomplish a constant temperature along the lines on either side of the conductor? This is the same problem that has been solved by the gradient method which, as noted [8], needs an alternative computational process because of the difficulties in constructing general purpose software yielding gradient information for the coupled problem.

Fig. 7b presents the associated boundary value problem formed from a quarter of the minimal system for analysis consisting of a square conductor (with $\mu_r = 10$ $\sigma_e = 15$ S/m, $\sigma_t = 0.1$ W/m/^oC). A current density $J_0 = 10 + j10$ A/m^2 has a relatively low frequency of $\omega = 10/s$, kept deliberately low to avoid a very fine mesh, our purpose here being to investigate and establish methodology rather than to solve large problems in their full complexity. The

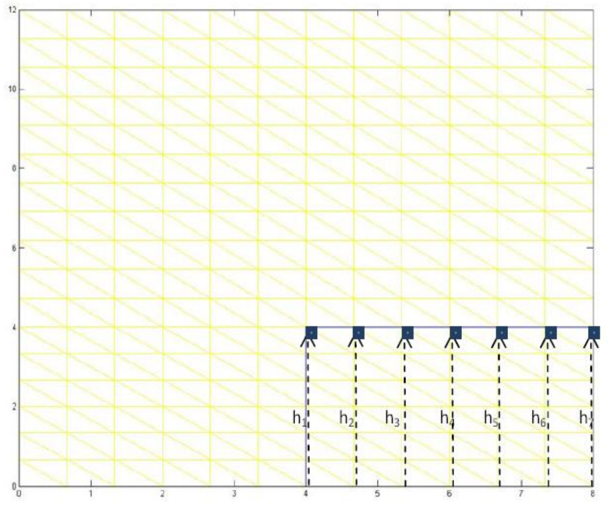


Figure 8: The Parameterized Geometry

top edge of the conductor has to be shaped to get a constant temperature profile of 60 ^{o}C, at $y = 6\ cm, and$ $4\ cm \leq x \leq 8\ cm$ as shown in Fig.7 along the measuring line where, along the lines of (6), we define the object function

$$F = \frac{1}{2}\sum_{i=1}^{10}[T_i - 60]^2 \tag{16}$$

where T_i is the temperature computed at the measuring point i. An erratic undulating shape arose when Pironneau optimized a pole face to achieve a constant magnetic flux density [35] and this was overcome through constraints [36]. Accordingly we extend the same principle, so as to maintain a non-undulating shape by imposing the constraints

$$h_1 > h_2 > h_3 > h_4 > h_5 > h_6 > h_7 \tag{17}$$

to ensure a smooth shape. The penalties were imposed by adding a penalty term to the object function F whenever it fails to satisfy the conditions for constraints [15, 23]

Radiation effects at the boundary are neglected, taking the boundary temperature to be the room temperature at 20 ^{0}C. As already stated, we avoid more exact details because our purpose here is to establish the feasibility of the methods we use, assessing the efficiencies to be gained by the use of GPU processing.

The parameterized problem specific mesh is shown in Fig. 8 where the device descriptive parameter set {p} consists of the 7 heights h_i. The numerical model was uniformly meshed with 234 nodes and 408 elements. This was deliberately kept crude to control debugging; after succeeding with the method and establishing that it works as a method and as programmed on the GPU, it was refined as necessary for higher accuracy.

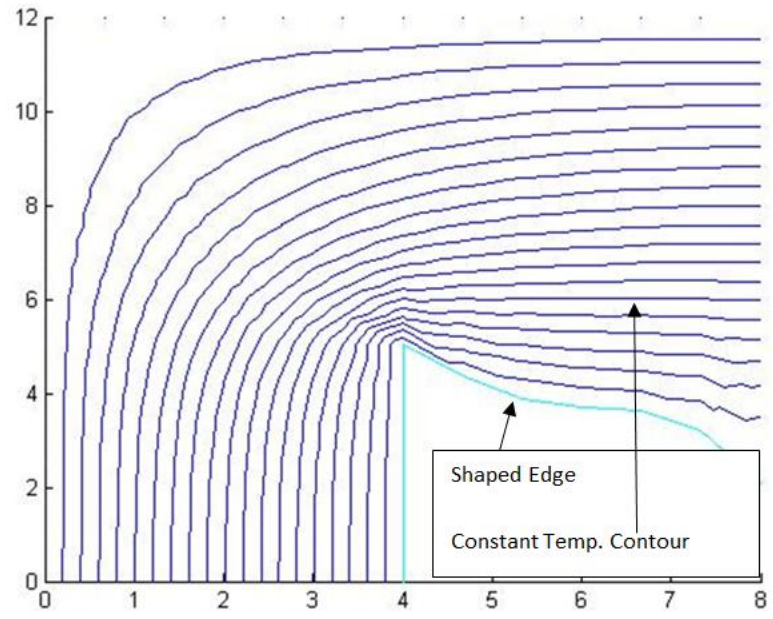


Figure 9: Optimal Shape by the Genetic Algorithm

In the process of optimization as these heights h_i change the mesh connections remain the same but the element sizes and shapes change. For the specific example shown in Fig. 8, the heights h_i are divided into six pieces so the locations of the seven equally spaced seven nodes along the height h_i would change. Accordingly the length from above the edge of the conductor being shaped to the vertical boundary will be adjusted and divided into 11 equal lengths.

The measuring line located at y = 6 cm, was sampled into 10 equally spaced points and tolerance boundaries of each $h_i$were set to

$$1.5\ cm \leq h\ \leq 5.5\ cm \tag{18}$$

Results and Discussion thereof

Fig. 9 shows the optimum shape of the conductor and temperature profile after 40 iterations for a population size of 512. The corner of the conductor rising toward the line where the constant 60 ^oC temperature is desired is as to be expected. For as seen in Fig. 10 (which shows the design goals being accomplished), the constant 60 ^oC temperature is perfectly matched. The lower graph giving the initial temperature shows that the temperature drops above the corner of the conductor. Therefore to address this, the corner has to rise close to the line of measurement to heat the line above the corner and Fig. 9 shows that this is what the optimization process has accomplished.

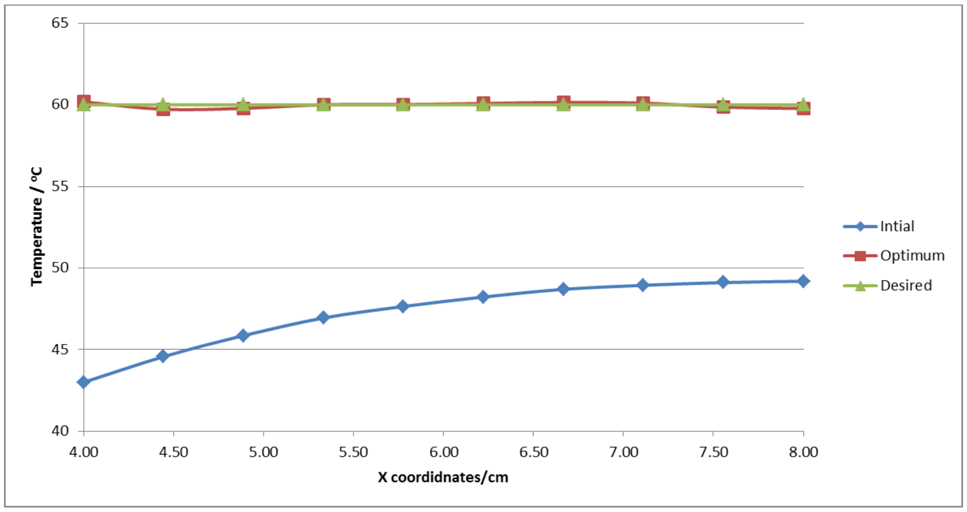


Figure 10: Temperature Distribution: Desired, Initial and Optimized

Significant speedup was accomplished with GPU computation as seen in Fig. 11. No gains in speedup beyond a factor of 28 were seen after a population size of 200. The meandering nature of the gain after that may be attributed to the happenstance inherent to a statistical method like the genetic algorithm.

The gain of 28 or so, however is lower than the "30 or more" reported by Cecka, Lew, and Darve [33] for their paper about the direct finite element solution paper but it is an impressive figure given the extensive communication time in a coupled problem such as this. To verify that the performance of our programming is superior we programmed the conjugate gradients solution for ever larger matrix sizes and measured the GPU:CPU ratio. Our findings are seen in Fig. 12 where we achieve a far superior gain of 147. This may be because Cecka *et al.* could achieve only a 2-3 fold gain in matrix assembly. Even accounting for that, however, their gain is very small considering that matrix assembly takes but trivial, negligible time in the finite element solution process where the preponderant computational load is from matrix solution. Our lower gains during optimization which includes matrix assembly is comparable to the accomplishments of Cecka *et al.* and needs further study as why such wide ranging gains occur.

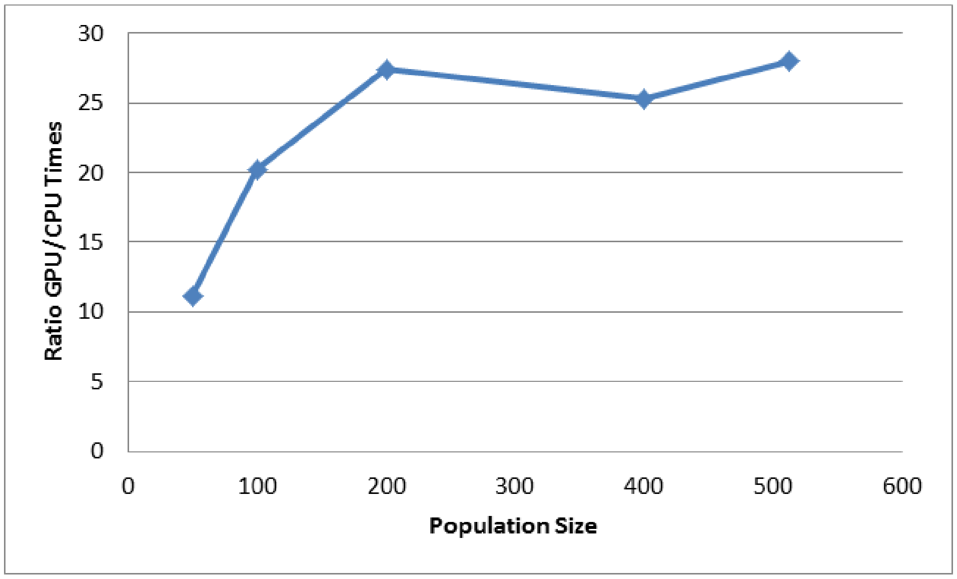


Figure 11: Speedup: GA Optimization GPU Time: CPU Time with

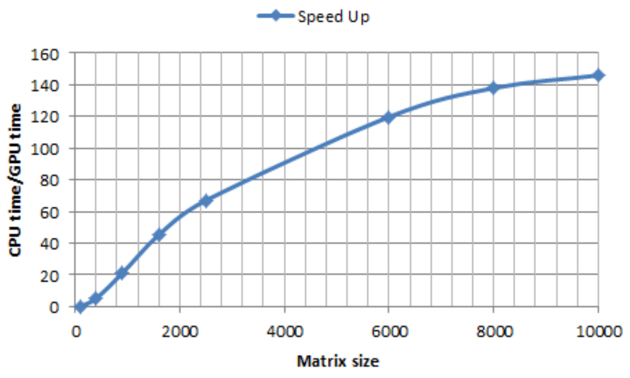


Figure 12: Speedup of the Conjugate Gradients Algorithm: Matrix Size Vs CPU time/GPU time

CONCLUSIONS

Shape optimization for the electroheat problem using GA has presented and validated using a simple geometry. This problem was also computed using GPU parallel computing techniques whereby speedups of 28 were demonstrated. This is comparable to the speedup of 30 recently demonstrated in the literature for a single finite element solution. Yet we have demonstrated a speedup of 148 for a single finite element matrix equation solution. A companion paper will show how such an immense speedup was achieved [37].

The next step to be taken is including thermal phenomena like radiation and convection and incorporating nonlinearity which could have implications to convergence. This model also has to be extended to 3-D.

Acknowledgements

This work was funded in part by the US Army's Tank, Automotive Research and Development Center (TARDEC) under Purchase Order No. SSP TCN 13057.

REFERENCES

[1] M.V.K. Chari, and S.J. Salon, *Numerical Methods in Electromagnetism*, Academic Press, 2000**.**

[2] L. Longeot, L. Nicolas, Ph. Wendling, "3D Design of an Inductor for Induction Heating Using 2D FEM and 3D BIEM Modeling," *IEEE Trans. Magn*., Vol. 27, No. 5, September 1991.

[3] A. A. Tseng, J. Zou, H. P. Wang, and S. R. H. Hoole, "Numerical Modeling of Macro and Micro Behaviors of Materials in Processing: A Review," J. of Computational Physics, Vol. 102, pp. 1 - 17, 1992.

[4] J. D. Lavers, "Numerical Solutions Methods for Electroheat Problems," *IEEE Trans. Magn*., Vol. 19, 1983.

[5] S. R. H. Hoole, V. Sathiaseelan and A. Tseng, "Computation of Hyperthermia-SAR Distributions in 3-D," IEEE *Trans. Magnetics*, Vol. MAG- 26, No. 2, pp. 1011-1014, March 1990.

[6] V. Sathiaseelan, B.B. Mittal, A.J. Fenn, and A. Taflove, "Recent advances in external electromagnetic hyperthermia," *Cancer treatment and Research*, Vol. 93, pp. 213-245, 1998.

[7] S. R. H. Hoole, *Computer-Aided Analysis and Design of Electromagnetic Devices*, Elsevier, New York, 1989.

[8] Tan. H. Pham and S. R. H. Hoole, "Unconstrained Optimization of Coupled Magneto-Thermal Problems," *IEEE Trans. Magn*., Vol. 31, No. 3, pp. 1988 - 1991 May 1995.

[9] S. R.Hoole (ed.), *Finite Elements, Electromagnetics and Design*, Elsevier, Amsterdam, May 1995.

[10] J.S. Arora and E.J. Hang, "Efficient Optimal Design of Structures by Generalized Steepest Descent Programming," *International Jo. For Num. Meth. Eng.*, Vol. 10, pp. 747-766, 1976.

[11] A. Marrocco and O. Pironneau, "Optimum Design with Lagrangian Finite Elements: Design of an Electromagnet," *Computer Methods in Applied Mechanics and Engineering,* Vol. 15, pp. 277-308, 1978.

[12] S. Gitosusastro, J.L. Coulomb and J.C. Sobonnadiere, "Performance Derivative Calculations and Optimization Process," *IEEE Trans. Magnetics*, Vol. 25(4), pp. 2834-2837, 1989.

[13] J.L. Coulomb, G. Meunier and J.C. Sabonnadiere, "An Original Stationary Method using Local Jacobian Derivative for Direct Finite Element Computation of Electromagnetic Force, Torque and Stiffness," *Journal of Magnetism and Magnetic Materials*, Vol. 26, pp. 337-339, 1982.

[14] S. R. H. Hoole, S. Subramaniam, Rodney Saldanha, J.-L. Coulomb, and J.-C. Sabonnadiere, "Inverse Problem Methodology and Finite Elements in the Identification of Inaccessible Locations, Sources, Geometry and Materials," *IEEE Trans. on Mag.*, Vol. 27, No. 3, pp. 3433-3443, May 1991.

[15] Garret N. Vanderplaats, *Numerical Optimization Techniques for Engineering Design:* McGraw-Hill, 1984.

[16] S. R. H. Hoole and P. R. P. Hoole, "On Finite Element Force Computation From Two- and Three-dimensional Magnetostatic Fields", *The J. of Applied Physics*, Vol. 57, No. 8, pp. 3850-3852, April 1985.

[17] S. R. H. Hoole, "Optimal Design, Inverse Problems and Parallel Computers," *IEEE Trans. Magn.*, Vol. 27, pp. 4146-4149, Sept. 1991.

[18] K. R. Weeber and S. R. H. Hoole, "Geometric Parametrization and Constrained Optimization Techniques in the Design of Salient Pole Synchronous Machines," *IEEE Trans. Magn.*, Vol. 28, pp. 1948-1960, July, 1992.

[19] Konrad Weeber and S. R. H. Hoole, "A Structural Mapping Technique for Geometric Parametrization in the Synthesis of Magnetic Devices," *Int. J. Num. Meth. Eng.* Vol. 33, pp. 2145-2179, July 15, 1992.

[20] J. Simkin and C.W. Trowbridge, "Optimization Problems in Electromagnetics," *IEEE Trans. Magnetics*, Vol. 27 (5), pp. 4016-4019, 1991.

[21] Randy L. Haupt and Sue E. Haupt, *Practical Genetic Algorithms*, John Wiley, 1997

[22] R. Haupt, "Comparison between Genetic and Gradient-based Optimization Algorithms for Solving Electromagnetics Problem," *IEEE Trans. Magnetics*, Vol. 31 (3), pp. 1932-1935, 1995.

[23] P. Venkataraman, *Applied optimization with MATLAB Programming*, 2nd edn., John Wiley, Hoboken, NJ, 2009.

[24] K. Preis, C. Magele and O. Biro, "FEM and Evolution Strategies in the Optimal Design of Electromagnetic Devices," *IEEE Trans. Magnetics*, Vol. 26(5), pp. 2181-2183, Sept. 1990.

[25] S. R. H. Hoole, K. Weeber and S. Subramaniam, "Fictitious Minima of Object Functions, Finite Element Meshes, and Edge Elements in Electromagnetic Device Synthesis," *IEEE Trans. Magn.*, Vol. 27, pp. 5214-5216, Nov. 1991.

[26] T.W. Manikas and J.T. Cain, "Genetic Algorithms vs. Simulated Annealing: A Comparison of Approaches for Solving the Circuit Partitioning Problem", Tech. Report TR-96-101, University of Pittsburgh, Dept. of Electrical Engineering, May 1996.

[27] S. R. H. Hoole, "Finite Element Electromagnetic Field Computation on the Sequent Symmetry 81 Parallel Computer," *IEEE Trans. on Magnetics*, Vol. MAG- 26, No. 2, pp. 837-840, March 1990.

[28] G. Mahinthakumar and S. R. H. Hoole, "A Parallel Conjugate Gradients Algorithm for Finite Element Analysis of Electromagnetic Fields," *J. of Applied Physics*, Vol. 67, No. 9, pp. 5818-5820, 1990.

[29] S. R. H. Hoole and G. Mahinthakumar, "Parallelism in Interactive Operations in Finite Element Simulation," *IEEE Trans. on Mag*., Vol. MAG-26, pp. 1252-1255, July 1990.

[30] G. Mahinthakumar and S. R. H. Hoole,"A Parallelized Element by Element Jacobi Conjugate Gradients Algorithm for Field Problems and a Comparison with other Schemes," *Int. J. App. Electromag. in Matl*., Vol. 1, No. 1, pp. 15-28, July 1990.

[31] S. R. H. Hoole, "A Review of Parallelization in Finite Element Device Simulation and Possiblities for Extensions," *Int. J. App. Electromag. in Matl.* Vol. 2, No. 1, pp. 99-108, 1991.

[32] S. R. H. Hoole and K. Agarwal, "Parallelization of Optimization Methods," IEEE Trans. Magn., Vol. 33, No. 2, pp. 1966-1969, March 1997.

[33] C. Cecka, A.J. Lew, and E. Darve, “Assembly of Finite Element methods on Graphics Processors,” *Int. J. Num. Meth. Eng.*, Vol. 85 (5), pp. 640–669, 2011

[34] http://on-demand.gputechconf.com/gtcexpress/2011/presentations/GTC_Express_Sarah_Tariq_June2011.pdf

[35] O. Pironneau, *Optimal Shape Design for Elliptic Systems*, Springer-Verlag, New York, 1984.

[36] S. Subramanaiam, A. A. Arkadan and S. R. H. Hoole, “Optimization of a magnetic pole face using linear constraints to avoid jagged contours “Constraints for Smooth Geometric Contours from Optimization,” *IEEE Trans. Magn*., Vol. 30 (5), pp. 3455-3458, 1994

[37] S. Sivasuthan, V.U. Karthik and S.R.H. Hoole, “An Assessment of Using the Graphics Processing Unit (GPU) for Parallelized Finite Element Field Computation” Under review.